(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平6－12395

(43)公開日　平成６年(1994)１月21日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/16	４３０	9190－5Ｌ		
	３２０　Ｋ	8840－5Ｌ		

審査請求　未請求　請求項の数７(全　７　頁)

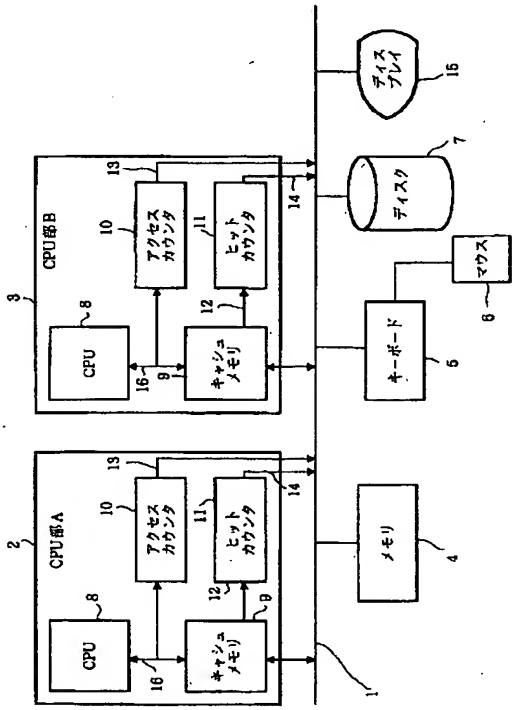
(21)出願番号　特願平4－170610

(22)出願日　平成４年(1992)６月29日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号
(72)発明者　榎田　幸
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内
(72)発明者　鈴木　茂夫
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内
(74)代理人　弁理士　大塚　康徳　(外１名)

(54)【発明の名称】　マルチプロセサシステムにおけるタスク割り付け方法

(57)【要約】
【目的】　マルチプロセサシステムにおいて、発生したタスクをプロセサに割り付ける際にプロセサによる処理効率が良い様に割り付けるプロセサを決定する。
【構成】　ＣＰＵ８が処理を行っている際、キャッシュメモリ９にアクセスすると、アクセスカウンタ１０によりアクセスの発生した回数が数えられる。また、発生したアクセスのうちで、所望の情報がキャッシュメモリ９内に存在していた回数をヒットカウンタ１１で数える。アクセスカウンタ及びヒットカウンタの値から、キャッシュメモリ９に所望の情報があった割合を求める。以上の手順をシステムを構成するＣＰＵすべてについて行い、前記割合の最も低いＣＰＵに新たに発生したタスクを割り付ける。



1

【特許請求の範囲】

【請求項１】　システムを構成するプロセサ各々について対となるキャッシュ記憶を備えたマルチプロセサシステムにおいて発生したタスクを前記プロセサに割りつけるタスク割り付け方法であって、

前記各プロセサが、前記対となるキャッシュ記憶へアクセスした回数を数える第１の計数行程と、

前記アクセスにより所望される情報が、前記対となるキャッシュ記憶に存在している回数を数える第２の計数行程と、

前記第１の計数工程による回数と前記第２の計数行程による回数とを基に、所望の情報がキャッシュ記憶に存在しているヒット率を算出する算出行程と、

前記ヒット率を基に前記発生したタスクを実行するプロセサを選択する選択行程と、

を備えることを特徴とするタスク割り付け方法。

【請求項２】　前記算出行程はタスクごとに前記ヒット率を算出することを特徴とする請求項１項記載のタスク割り付け方法。

【請求項３】　前記選択行程は、前記各プロセサにおいて前記ヒット率の最も高いタスクのうち、前記ヒット率が最も低いタスクを実行するプロセサを選択することを特徴とする請求項２項記載のタスク割り付け方法。

【請求項４】　前記算出行程はプロセサごとに前記ヒット率を算出することを特徴とする請求項１項記載のタスク割り付け方法。

【請求項５】　前記選択行程は前記ヒット率が最も低いプロセサを選択することを特徴とする請求項４項記載のタスク割り付け方法。

【請求項６】　前記ヒット率の算出はタスクの開始から終了までの時間内のアクセスを対象としてなされることを特徴とする請求項１項記載のタスク割り付け方法。

【請求項７】　前記ヒット率の算出は適当に選ばれた一定の時間内のアクセスを対象としてなされることを特徴とする請求項１項記載のタスク割り付け方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、例えばＣＰＵの動作状態を考慮してタスクの割り付けを行なうマルチプロセサシステムに関するものである。

【０００２】

【従来の技術】従来、複数のＣＰＵを有するコンピュータシステム（以下マルチプロセサシステムと呼ぶ）では、１つのマスタとなるＣＰＵとそれに従属するスレーブのＣＰＵとから構成されている場合には、各ＣＰＵの処理内容が決められていた。また、各ＣＰＵが同格の場合には、空いているＣＰＵがあれば新たに実行すべきタスクをそこに割り付けていた。このとき、もし全ＣＰＵが処理中であれば、新規に処理すべきタスクの優先順位と実行中のタスクの優先順位との比較結果から実行タスクを切り替えたり、比較したタスクの優先順位が同じ場合には、強制的にいずれかのＣＰＵに割り当て、タイムシェア方式により複数のタスクを切り替えながら実行する様に構成されていた。

2

【０００３】

【発明が解決しようとする課題】しかしながら、上記従来例のマスタスレーブ構成のマルチプロセサシステムでは、あるタスクを実行する場合に、処理内容によるＣＰＵの割り当てのために、空いているＣＰＵがあったとしても、必ず特定のＣＰＵで実行せねばならないことがある。例えばインタラプト処理のようなタスクの場合には、スレーブＣＰＵが空いていても必ずマスタＣＰＵで処理せねばならず、マスタＣＰＵの負荷が増して、システム全体のスループットが落ちると言う欠点があった。

【０００４】一方、各ＣＰＵが同格なマルチプロセサシステムにおいて、全ＣＰＵが処理を行なっている時に別のタスクの実行を新たに要求された場合には、実行中のタスクの優先順位と新規のタスクの優先順位とが同じであると、例えば各ＣＰＵについているデバイス番号の順序といった所定の規則に従ってＣＰＵを割り当てる。新規のタスクは割り当てられたＣＰＵ上で既に実行しているタスクとタイムシェアして実行される。このため、ＣＰＵの動作状態に関係なく割り当ててしまい、タイムシェア方式によるタスク制御を行なうので、新規タスクのＣＰＵへの割り付けは処理効率についての考慮がなく、非効率的な割り当てがなされる得るという欠点があった。

【０００５】本発明は上記従来例に鑑みてなされたもので、ＣＰＵの動作状態を参照して、ＣＰＵを効率よく使用できる様にタスクの割りつけを決定できるタスク割り付け方法を提供することを目的とする。

【０００６】

【課題を解決するための手段】上記目的を達成するために、本発明のマルチプロセサシステムにおけるタスク割り付け方法は次のような構成からなる。

【０００７】システムを構成するプロセサ各々について対となるキャッシュ記憶を備えたマルチプロセサシステムにおいて発生したタスクを前記プロセサに割りつけるタスク割り付け方法であって、前記各プロセサが、前記対となるキャッシュ記憶へアクセスした回数を数える第１の計数行程と、前記アクセスにより所望される情報が、前記対となるキャッシュ記憶に存在している回数を数える第２の計数行程と、前記第１の計数工程による回数と前記第２の計数行程による回数とを基に、所望の情報がキャッシュ記憶に存在しているヒット率を算出する算出行程と、前記ヒット率を基に前記発生したタスクを実行するプロセサを選択する選択行程とを備える。

【０００８】

【作用】上記構成により、マルチプロセサシステムを構成する各プロセサがキャッシュメモリをアクセスした際

の、所望の情報がキャッシュメモリ上に存在している率を算出し、新たに発生したタスクを割りつけるプロセサをその率に応じて決定する。

【０００９】

【実施例】

［実施例１］本発明の第１の実施例の構成を図１に示す。本実施例では、説明を簡単にするために、ＣＰＵ部を２個持ったマルチプロセサシステムを説明する。

【００１０】＜構成＞図中１はシステム内のＣＰＵやメモリ等の資源を接続するためのシステムバスであり、２はＣＰＵ部Ａ、３はＣＰＵ部Ｂである。４は各ＣＰＵ部から共通にアクセスできるメモリ部、５はコマンド等を入力するためのキーボード、６は同じく入力のためのマウス、７はプログラムやデータを格納するためのディスク、１５は表示部である。ＣＰＵ部ＡとＣＰＵ部Ｂとは同じ構成をしており、図中、同じ構成要素には同じ番号をつけてある。ＣＰＵ部Ａについてその構成を説明する。

【００１１】ＣＰＵ部Ａは、ＣＰＵ８と、メモリアクセスを高速に行なうためにＣＰＵ８のローカルなデータを格納しておくキャッシュメモリ９と、ＣＰＵ８からのメモリアクセス信号１６によってキャッシュメモリへのアクセス回数をカウントするアクセスカウンタ１０と、ＣＰＵ８からのメモリアクセス要求に対してキャッシュメモリ９内に該当データがあったことを意味するヒット信号１２をとり、キャッシュ・ヒット回数をカウントするヒットカウンタ１１とから構成される。カウンタ１０の値とカウンタ１１の値とはシステムバス１上にマッピングされ、各ＣＰＵからアクセスできる様にされている。キャッシュメモリ９は、ＣＰＵ８からのメモリアクセス要求に対してデータが存在していればヒット信号１２を出し、その値をＣＰＵ１に出力する。もし存在していなければキャッシュメモリ９はヒット信号１２を出さず、メモリ４またはディスク７から該当するデータをキャッシュメモリ９にローディングし、ＣＰＵ８に求められたデータを返す。またアクセスカウンタ１０とヒットカウンタ１１とは、ＣＰＵに新しいタスクを割り付けた時に共に０にクリアされ、図示していない各ＣＰＵ部のタイマ等の手段により、所定時間経過ごとに０クリアされる。

【００１２】＜タスクの割り付け＞以上のような構成のマルチプロセサシステムにおいて、ＣＰＵ部Ａでタスク Ａが、ＣＰＵ部Ｂでタスク Ｂが実行されている時に、キーボード５やマウス６からの入力でユーザが要求により、あるいは、図示しない外部からの割込みにより新たにタスクＣを実行しなければならない場合を考える。

【００１３】図２にタスクＡとタスクＢの各カウンタの変化例を示す。図２によれば、２０は、時刻t_1におけるＣＰＵ部Ａのアクセスカウンタ１０が１００、ヒットカウンタ１１が５０であることを示し、ヒット率は５０／１００＝５０％であることが分かる。同様にＣＰＵ部Ｂのヒット率は５０／１５０≒３３％である。更に時間が経過し、新たなタスクＣの実行要求が発生した時の処理を、図６のフローチャートを参照して説明する。

【００１４】時刻t_2にタスクＣの実行要求が発生すると、まずその時点で遊んでいるＣＰＵの有無をテストし、空いているＣＰＵがあればそのＣＰＵで新たに発生したタスクを実行する。すべてのＣＰＵ（この場合ＣＰＵ部ＡとＣＰＵ部Ｂ）で何らかのタスクが処理されているなら、それらの中から最適なＣＰＵを選び出さねばならない。

【００１５】まず、ＣＰＵ部ＡのＣＰＵ８がアクセスカウンタ１０とヒットカウンタ１１の値を読み（Ｓ６０１・Ｓ６０２）、ヒット率を計算する（Ｓ６０３）。ＣＰＵ部Ａでのキャッシュ・ヒット率は表より１５０／２００＝７５％となる。この値はキャッシュメモリ９に格納しておき、次にＣＰＵ部Ｂについてヒット率を計算する（Ｓ６０４－ＮＯ）。ＣＰＵ部Ａと同じ要領で計算すると、ＣＰＵ部２のキャッシュヒット率は２７０／３００＝９０％となり、これもキャッシュメモリ９に格納しておく。これですべてのＣＰＵについてヒット率の計算が済んだ（Ｓ６０４－ＹＥＳ）。次に各ＣＰＵ部のヒット率７５％と９０％とを比較し、小さいヒット率の方を新規タスクを割り当てるＣＰＵと決定する（Ｓ６０５）。最後に小さいヒット率のＣＰＵすなわちＣＰＵ部Ａにタスク Ｃを割り付ける（Ｓ６０６）。これは、キャッシュヒット率の高い方では、キャッシュメモリへの外部の記憶装置からの読み込みが少なく、より高い効率でＣＰＵが稼働している。一方ヒット率の低い方ではキャッシュメモリへの外部記憶からのロードがより頻繁であるため、複数のタスクをタイムシェアして実行する際のオーバーヘッドがヒット率のより高いものほどには全体の処理効率に影響を与えない。このため、高効率の方を優先して処理を進められるように、ヒット率の低いＣＰＵへと新たなタスクを割りつける。

【００１６】これにより、ＣＰＵ部Ａはタスク Ａとタスク Ｃを各タスクの優先順位によりタイムシェア方式により実行し、ＣＰＵ部Ｂはそのままタスク Ｂを実行する。

【００１７】【他の実施例】第１の実施例では、各ＣＰＵ部がタスクを１つずつ実行している場合を説明したが、本実施例では、各ＣＰＵ部が既に複数のタスクを実行している場合を考えてみる。この場合には、ＣＰＵ部Ａでタスク ＡとタスクＢとが、ＣＰＵ部Ｂでタスク ＣとタスクＤとが実行されているものとし、ここにタスクＥの実行要求が発生した場合を説明する。

【００１８】［実施例２］第２の実施例では、各ＣＰＵ部で現在いくつのタスクが実行中であるかを意識しないで、各ＰＣＵ部ごとのキャッシュ・ヒット率に従って、どのＣＰＵにタスクＥを割り付けるかを決定する方法を

説明する。この場合は、第1の実施例と同様に、図3の表に示す様に各ＣＰＵ部毎のメモリアクセス回数と、キャッシュ・ヒット回数とを図1に示す構成で記憶していけば良く、またタスクを割りつけるＣＰＵ部を決定する方法も第1の実施例に示した方式で決定すれば良い。

【００１９】図3ではＣＰＵ部ＡとＣＰＵ部Ｂとについてアクセスカウンタとヒットカウンタとが記録されており、ヒット率は実施例1と同じ手順で導くことができ、タスクの割り付け先はＣＰＵ部Ａと決まる。

【００２０】［実施例3］第3の実施例では、各タスク毎のキャッシュ・ヒット率を計算し、それから新しいタスクをどのＣＰＵ部に割り付けるかの方法を説明する。図4に本実施例の構成図を示す。図4中で図1と同じ動作をする構成要素には同じ番号をつけてある。３０のカウンタ制御部は、各ＣＰＵ部にあるメモリアクセス回数用カウンタであるアクセスカウンタ１０と、キャッシュ・ヒットの回数用カウンタであるヒットカウンタ１１とから出力されるカウント情報を各々信号線３３・信号線３４から入力し、各ＣＰＵ部のカウント情報を記憶する。さらに各ＣＰＵから、各ＣＰＵで実行タスクを切り替えたことを知らせる信号３５も入力する。カウンタメモリ３２は、各ＣＰＵで実行中の全タスクのメモリアクセス回数とキャッシュヒット回数を蓄積するためのメモリである。

【００２１】以下、詳細な説明をするが、説明のため、ｕｎｉｘ等のＯＳで用いられているタスクを識別するためのユニークな識別子（以下タスクＩＤと呼ぶ）を用いてタスクを識別する。実行中の４タスクについて、タスクＡは１００、タスクＢは１０２、タスクＣは１０１、タスクＤは１０３とする。カウンタコントローラ３１は、システム起動時にはすべてのＣＰＵについて、各ＣＰＵからカウンタ情報が入力された時にはそのＣＰＵについて、アクセスカウンタ１０、ヒットカウンタ１１を０クリアする。タスクＡがＣＰＵ１で実行され、所定時間経過すると、ＣＰＵ１はカウンタコントローラ３１に対して自分のＣＰＵ番号と実行中のタスクＩＤ、カウント情報であるアクセスカウンタ１０とヒットカウンタ１１の値を送る。タスクＢ・Ｃ・Ｄについても同様な処理を行う。そうすると、カウンタコントローラ３１は、カウンタメモリ３２上に図5に示すような、或る時刻におけるカウンタを記録したテーブル情報を作成する。図4のアクセスカウンタ１０、ヒットカウンタ１１のフィールドの意味は第1の実施例と同じである。ここで図4の状態で、タスクＥの実行要求が発生したとする。そうすると、カウンタコントローラは各タスク毎にキャッシュ・ヒット率を計算する。すなわち、

タスク１００：　６０／１００＝６０％　…　ＣＰＵ部Ａ

タスク１０２：１４０／２００＝７０％　…　ＣＰＵ部Ａ

タスク１０１：２４０／３００＝８０％　…　ＣＰＵ部Ｂ

タスク１０３：　６０／２００＝３０％　…　ＣＰＵ部Ｂ

を求める。ここで、４つのキャッシュ・ヒット率の中で1番高いもの、この場合タスク１０１の８０％を見つけ、効率の良いタスクを先に実行するという観点から、タスク１０１はＣＰＵ部Ｂで実行しているので、ＣＰＵ部ＡにタスクＥを割り付ける旨を判断し、タスクＥをＣＰＵ部Ａに割り付ける。これは、実施例1の手順をＣＰＵ単位からタスク単位へと変更したものである。従って、図6のフローチャートのステップＳ６０５を「ヒット率の高いタスクを実行しているＣＰＵを順に除外して、最後に残ったＣＰＵを選ぶ」とすれば、本実施例の適用できる。

【００２２】こうして実現されたタスク割り付け方法では、各ＣＰＵが実行中のタスクの実行状態として、ＣＰＵとキャッシュ間のキャッシュ・ヒット率をモニタする手順を設けることにより、別タスクの実行要求が発生した時に、各ＣＰＵで実行中のタスクの優先順位が同じ場合には、キャッシュ・ヒット率の一番低いＣＰＵに今回のタスクを割り付けることに、キャッシュ・ヒット率の高いタスクはそのまま高速に処理を続けることができ、キャッシュ・ヒット率の低いＣＰＵでタイムシェア方式等により処理することにより、このキャッシュ・ヒット率の低いタスクは、もともとキャッシュメモリにあまりヒットしないメモリ・アクセスをしているためにタスクスイッチによるキャッシュ・ヒット率の低下が少なく、効率良くＣＰＵ等の資源を割り付けができる。また、キャッシュ・ヒット率を求めるためのハード規模も小さくすむ。

【００２３】これまでの説明では、ＣＰＵのタイマ等により所定の単位時間を測定し、その単位時間内のキャッシュ・ヒット率を計算する様に説明してきたが、そのタスクの開始から終了まで通したキャッシュ・ヒット率を求める方式でも良い。例えば、大容量の画像データを順次処理して行くようなタスクの場合には、そのタスク全体としてみればキャッシュ・ヒット率は小さいものと考えられるし、また元々時間のかかる処理であるため、他のタスクとタイムシェアして実行しても構わない。

【００２４】さらに、キャッシュ・ヒット率を第1・第2の実施例ではＣＰＵ部Ａ２で、第3の実施例ではカウンタ制御部３０で求めていたが、これに限るものでもない。

【００２５】また、これまでの実施例の説明ではＣＰＵ部は2個で説明したが、これに限るものでもなく、更に多くのＣＰＵを持った装置であっても良い。また、各ＣＰＵ上で実行しているタスク数が同じ場合を説明したが、これに限るものでもなく、空いているＣＰＵがあれば、そのＣＰＵに割り付けたりすることは容易に考えら

れる。また、これまではキャッシュ・ヒット率の高いタスクを優先的に実行する様に説明したが、これに限るものでもなく、キャッシュが余りヒットしないタスクは時間がかかるためにそれを優先して実行するという、これまで説明した実施例とは逆の制御方式も容易に考えられる。

【００２６】尚、本発明は、複数の機器から構成されるシステムに適用しても１つの機器から成る装置に適用しても良い。また、本発明は、システム或は装置にプログラムを供給することによって達成される場合にも適用できることはいうまでもない。

【００２７】

【発明の効果】以上説明した様に本発明に係るタスク割り付け方法は、ＣＰＵの動作状態を参照して、ＣＰＵを効率よく使用できる様にタスクの割りつけを決定できる。

【図面の簡単な説明】

【図１】第１の実施例のブロック図である。

【図２】第１の実施例のキャッシュヒット率を算出するための表である。

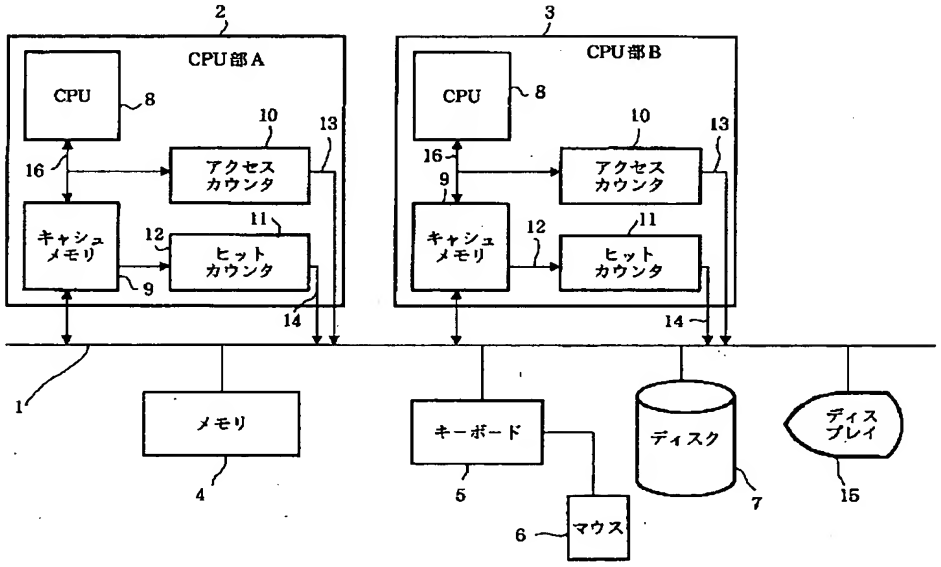
【図３】第２の実施例のキャッシュヒット率を算出するための表である。

【図４】第３の実施例のブロック図である。

【図５】第３の実施例のキャッシュヒット率を算出するための表である。

【図６】第１の実施例のフローチャートである。

【図１】



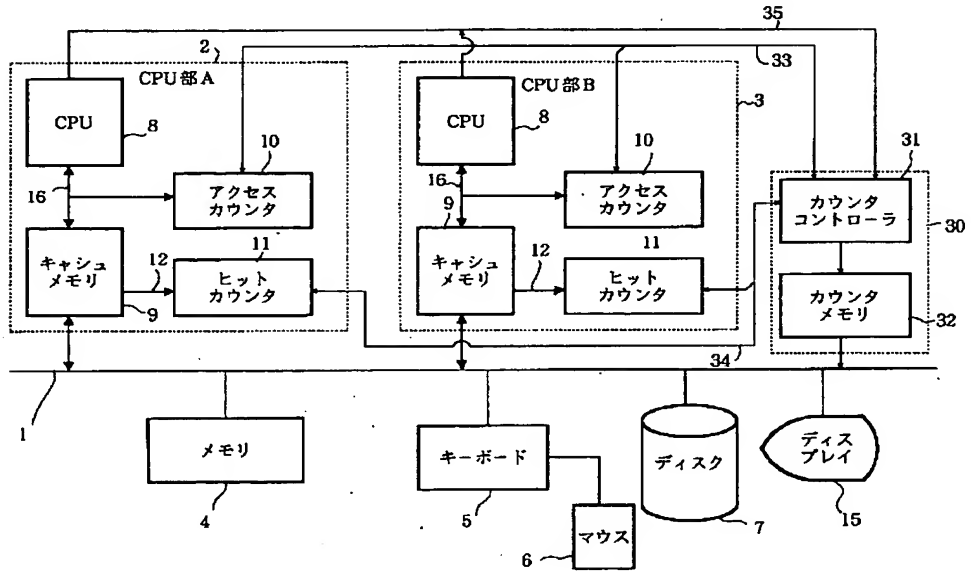
【図２】

プロセス名／カウンタ名／時刻	プロセスA		プロセスB		
	アクセスカウンタ	ヒットカウンタ	アクセスカウンタ	ヒットカウンタ	
t_1	100	50	150	50	20
t_2	200	150	300	270	21

【図３】

CPU名 / カウンタ名 / 時刻	CPU A		CPU B	
	アクセスカウンタ	ヒットカウンタ	アクセスカウンタ	ヒットカウンタ
t_1	300	200	200	300
t_2	・・・	・・・	・・・	・・・

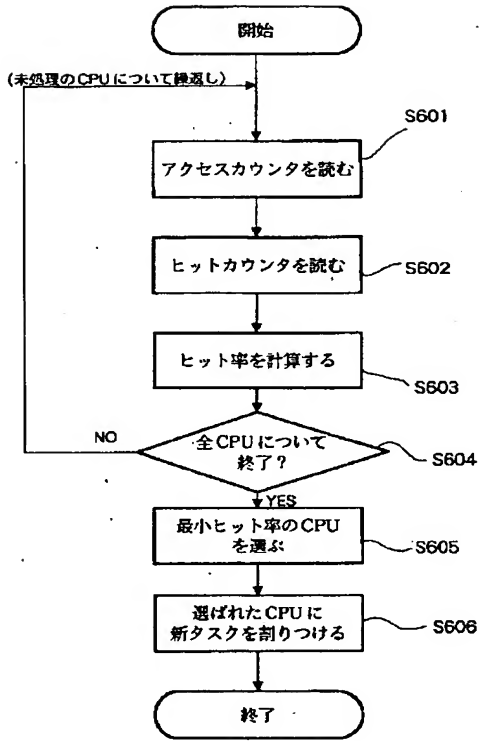
【図４】



【図５】

CPU部A			CPU部B		
	アクセスカウンタ	ヒットカウンタ		アクセスカウンタ	ヒットカウンタ
100	100	60	101	300	240
102	200	140	103	200	60

【図６】


開始
(未処理のCPUについて繰返し)
S601
アクセスカウンタを読む
ヒットカウンタを読む
S602
ヒット率を計算する
S603
NO
全CPUについて終了？
S604
YES
最小ヒット率のCPUを選ぶ
S605
選ばれたCPUに新タスクを割りつける
S606
終了